

Readme

National Center for Biotechnology Information Basic Local Alignment Search Tool (NCBI BLAST) HPC Sample

Lab version: 1.0.0

Last updated: 8/30/2011

**Contents**

[Overview 3](#_Toc297639868)

[Getting Started 5](#_Toc297639869)

[Task 1 - Inspecting the Blast Solution 5](#_Toc297639870)

[Task 2 - Inspecting the Parametric Sweep Script Files 6](#_Toc297639871)

[Task 3 - Preparing the Blast Solution for Deployment 8](#_Toc297639872)

[Task 4 - Uploading Input Files to Blob Storage 9](#_Toc297639873)

[Deployment 12](#_Toc297639874)

[Task 1 – Deploying the Parametric Sweep Application to Windows Azure Nodes 12](#_Toc297639875)

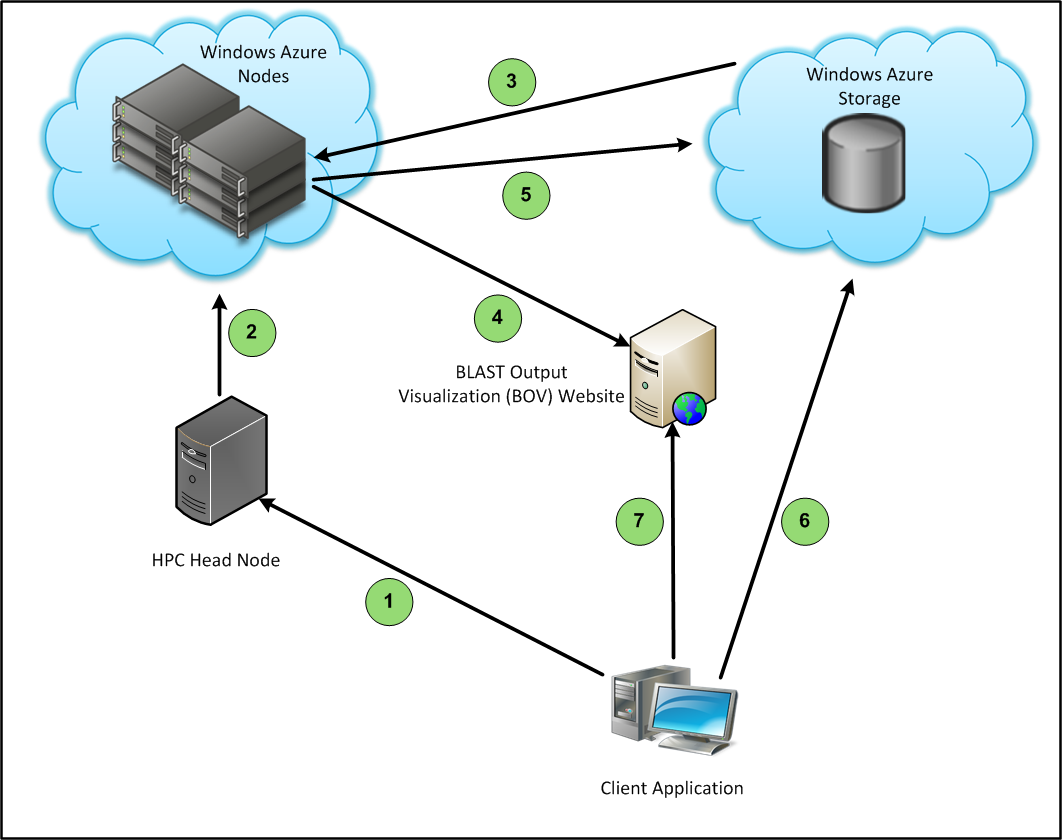
[Running the Client 14](#_Toc297639876)

[Task 1 – Verifying Cluster State 14](#_Toc297639877)

[Task 2 - Running the Client Application 14](#_Toc297639878)

[Summary 18](#_Toc297639879)

Overview

* 1. The National Center for Biotechnology Information’s Basic Local Alignment Search Tool (NCBI BLAST) HPC Sample demonstrates how to run a [nucleotide match algorithm](http://blast.ncbi.nlm.nih.gov/Blast.cgi) on the human genome using an HPC parametric sweep application.
  2. The parametric sweep application uses a set of input files that contain sequences of nucleotides, comparing them to the human genome database. The application creates output files containing sequence similarities and uploads these files to a BLAST output visualizer (BOV) website.
  3. To run the nucleotide match, the sample uses the blastn utility, which is a part of the [BLAST+](http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download) application.
  4. The architecture of the sample and the steps of its execution are described in Figure 1:
  5. 
  6. Figure 1
  7. Architecture of the BLAST sample
  8. The client application submits a parametric sweep job to the HPC Job Scheduler using the representational state transfer (REST) interface of the HPC Pack web features.
  9. The Windows HPC Server 2008 R2 SP2 cluster submits the job to the Windows Azure nodes.
  10. Each parametric sweep application downloads an input file from a Windows Azure blob storage. The input file includes a nucleotide that is compared to the human genome database previously downloaded to each Windows Azure compute node.
  11. After completing a sweep index, the BLAST application uploads the resulting matches file to the BLAST output visualization (BOV) website and receives a matching URL for the file’s visualization page.
  12. The output file and the URL are written to Windows Azure storage: the file is uploaded to a blob, and the URL is written in a table.
  13. While the parametric sweep job is running, the client application retrieves the list of URLs from Windows Azure table storage and shows it to the user.
  14. The user can select any of the URLs to see the rendered image for the nucleotide match.
  15. **Note:** To use this sample application, you will need to download the human genome compressed database from the NCBI FTP server, extract the database, and copy it to a Windows Azure blob storage as described later on in the document.

1. This sample demonstrates some of the new features offered by Windows HPC Server 2008 R2 Service Pack 2 (SP2). Refer to the [What's New in Windows HPC Server 2008 R2 Service Pack 2](http://technet.microsoft.com/en-us/library/hh184314(WS.10).aspx) article on TechNet for the complete list of new features offered in this version.

# Key Features

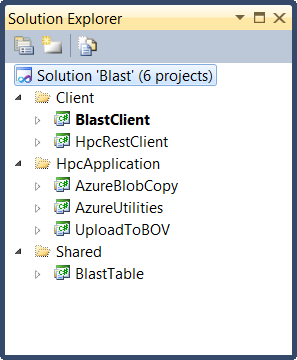
* 1. This sample demonstrates the following features:
  + Uploading a parametric sweep application package to Windows Azure nodes.
  + Uploading and downloading files to Windows Azure blob storage.
  + Writing and reading from Windows Azure table storage.
  + Creating and running a parametric sweep job in a Windows HPC Server 2008 R2 SP2 cluster.
  + Controlling jobs with the REST interface of the HPC Pack web features.

Getting Started

* 1. The following prerequisites are required for running the sample:
  + On the HPC Cluster’s head node:
    - [HPC Pack 2008 R2 SP2](http://www.microsoft.com/download/en/details.aspx?id=15368)
    - [HPC Pack web features](http://technet.microsoft.com/en-us/library/hh314627(WS.10).aspx)
      1. **Note:** When configuring the web components according to the [Installing the Microsoft HPC Pack Web Components](http://technet.microsoft.com/en-us/library/hh314627(WS.10).aspx) article, follow the steps for using the Basic authentication method.
  + On the Development machine:
    - [Microsoft Visual Studio 2010](http://www.microsoft.com/visualstudio/en-us)
    - [Windows Azure SDK and Windows Azure Tools for Microsoft Visual Studio (March 2011)](http://www.microsoft.com/download/en/details.aspx?id=15658)
    - [HPC Pack 2008 R2 SP2 Client Utilities](http://www.microsoft.com/download/en/details.aspx?id=17017)
  1. To run this sample, you need to have administrative access to your HPC cluster’s head node, and you must have a valid Windows Azure account, a Windows Azure worker node template defined in your head node, and several Windows Azure worker nodes in the HPC cluster that are started and online. Follow the [Deploying Windows Azure Worker Nodes in Windows HPC Server 2008 R2 Step-by-Step Guide](http://technet.microsoft.com/en-us/library/gg481749%28WS.10%29.aspx) on TechNet for further information.
  2. **Note:** The BLAST HPC parametric sweep application runs only one instance of the application on each Windows Azure node at a time, using only one core for the processing. To conserve Windows Azure CPU hours, it is advisable that you use only small nodes for this sample. If you wish to run this sample on medium or larger nodes, you can do so, but the application will only take advantage of a single core in each node.
  3. One possible design to utilize all Cores on larger instances is a file-based locking mechanism, of which each core has its own copy of the database.  When each new task is dispatched onto a node, it first checks to see whether one of the databases is available, claims it, and releases the claim when the task is done.  This is left as an additional exercise for this lab.

Task 1 - Inspecting the Blast Solution

In this task, you will inspect the **Blast** solution to see the various projects included in this sample.

* 1. Open Microsoft Visual Studio 2010 from **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Microsoft Visual Studio 2010**.
  2. Open the ***Blast.sln*** solution file located in the ***Blast\Source\Blast*** folder.
  3. Examine the projects tree in the **Solution Explorer** window. The solution tree includes several Windows Azure projects, some class libraries, and a console application, as shown in Figure 2:
     1. 
     2. Figure 2
     3. The Blast projects tree
  4. The following list describes the purpose of each project in the **Blast** solution:
  + Client
    - **BlastClient**. The project contains a Windows Presentation Foundation (WPF) application that submits the BLAST HPC job and monitors it. This project uses the **HpcRestClient** library to send job creation requests over HTTP to the HPC Job Scheduler’s web service using the REST interface.  
      You can inspect the **MainWindow.xaml.cs** file in this project to see how the WPF application uses the **HpcRestClient** library to create the new parametric sweep job.
    - **HpcRestClient**. The project contains a library that supports the REST interface of the HPC Job Scheduler’s web service, which is a part of the HPC Pack web features. Using this library, clients can send REST requests over HTTP to the HPC Job Scheduler’s web service in order to create new parametric sweep jobs and get information about active and finished jobs.  
      You can inspect the **HpcClient.cs** file in this project to see how the REST interface is used and which data structures the interface receives and returns.
  + HpcApplication
    - **AzureBlobCopy**. The project contains a command-line utility that enables users to upload and download files from Windows Azure blob storage.
    - **AzureUtilities**. The project contains a library for handling the uploading and downloading of files from blobs. **AzureBlobCopy** makes use of this library for that purpose.  
      You can inspect the **BlobUtilities.cs** file in this project to see how the library uses the **Microsoft.WindowsAzure.StorageClient** assembly to communicate with blobs.
    - **UploadToBOV**. The project contains a command-line utility that uploads BLAST output files to the BLAST output visualization (BOV) website, gets the generated URL for the file, and writes it to Windows Azure table storage.
  + Shared
    - **BlastTable**. The project contains a library for handling the writing and reading of blast results from a table storage. Both **UploadToBOV** and **BlastClient** make use of this library for that purpose.  
      You can inspect the **TableHelper.cs** file in this project to see how the library uses the **Microsoft.WindowsAzure.StorageClient** assembly to communicate with tables.

1. Task 2 - Inspecting the Parametric Sweep Script Files
   1. In this task, you will inspect the contents of the run.cmd and the verifydb.cmd files to see which commands run when the parametric sweep job executes in each Windows Azure compute node.
   2. Open the labs folder in **Windows** **Explorer** and navigate to the **BLAST\Source\ncbi-blast-2.2.25+** folder.
   3. Locate the verifydb.cmd file and view its contents:
      1. VERIFYDB.CMD
      2. set root=%CCP\_PACKAGE\_ROOT%\ncbi-blast-2.2.25
      3. set dbdir=%CCP\_PACKAGE\_ROOT%\ncbidb
      4. if exist %dbdir%\\*.\* goto finish
      5. if not exist %dbdir% mkdir %dbdir%
      6. echo copying files
      7. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nhr
      8. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nin
      9. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nnd
      10. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nni
      11. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nog
      12. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nsd
      13. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nsi
      14. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.00.nsq
      15. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nhr
      16. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nin
      17. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nnd
      18. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nni
      19. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nog
      20. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nsd
      21. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nsi
      22. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.01.nsq
      23. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer ncbi -LocalDir %dbdir% -FileName est\_human.nal
      24. :finish

The verifydb.cmd file runs in each of the Windows Azure nodes, as part of a preparation task, before starting the parametric sweep task. The script performs the following commands:

* + 1. Verifies the existence of the NCBI human genome database in the Windows Azure node.
    2. If the database is not present, runs the AzureBlobCopy utility to download the entire database into the *ncbidb* folder.
  1. Locate the **run.cmd** file and view its contents:
     1. RUN.CMD
     2. set inputFile=input\_%1
     3. set outputFile=output%1.txt
     4. set BLASTDB=%CCP\_PACKAGE\_ROOT%\ncbidb
     5. set root=%CCP\_PACKAGE\_ROOT%\ncbi-blast-2.2.25
     6. set dbdir=%CCP\_PACKAGE\_ROOT%\ncbidb
     7. set inputdir=%CCP\_WORKDIR%\%CCP\_JOBID%\%CCP\_TASKID%\input
     8. set outputdir=%CCP\_WORKDIR%\%CCP\_JOBID%\%CCP\_TASKID%\output
     9. if not exist %inputdir% mkdir %inputdir%
     10. if not exist %outputdir% mkdir %outputdir%
     11. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Download -BlobContainer inputncbi -LocalDir %inputdir% -FileName %inputFile%
     12. %root%\bin\blastn.exe -db est\_human -query %inputdir%\%inputFile% -out %outputdir%\%outputFile%
     13. %root%\AzureBlobCopy\AzureBlobCopy.exe -Action Upload -BlobContainer outputncbi -LocalDir %outputdir% -FileName %outputFile%
     14. %root%\UploadToBOV\UploadToBOV.exe %outputdir%\%outputFile%
     15. rem cleanup
     16. rmdir /S /Q %inputdir%
     17. rmdir /S /Q %outputdir%

The run.cmd file executes the following commands:

* + 1. Runs the AzureBlobCopy utility to download the current nucleotide input file from the input blob and decompress it.
    2. Runs the blastn application, which performs the nucleotide matching with the NCBI human genome database.
    3. Runs the AzureBlobCopy utility to upload the resulting file to the output blob.
    4. Runs the UploadToBOV utility to upload the resulting file to the BOV website, and writes the URL returned by the website to Windows Azure table.
    5. Clears the input and output files to conserve space on the Windows Azure node.

Task 3 - Preparing the Blast Solution for Deployment

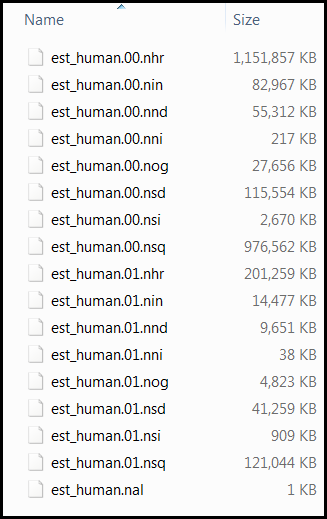
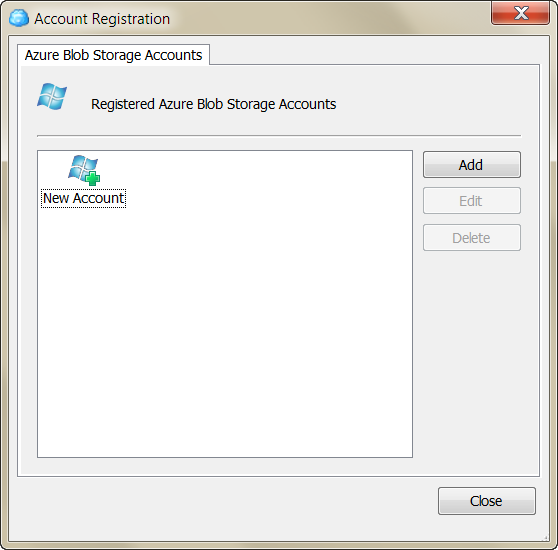
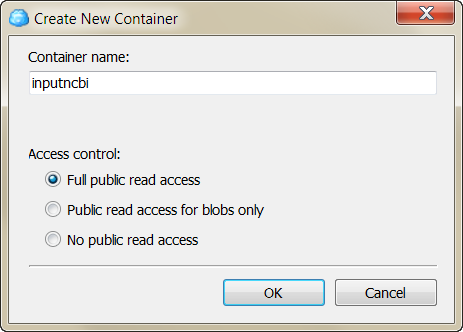
In this task, you will make the necessary adjustments to the projects’ configuration files so you can deploy them to the Windows Azure nodes.

* 1. In the **Solution Explorer** window, expand the **Client** folder, expand the **BlastClient** project, and open the ***app.config*** file. The **appSettings** section contains four keys: ***StorageAccountName***, ***StorageKey***, ***HpcWebServiceUrl***, and ***NodesGroup***.
     1. **StorageAccountName** and **StorageKey**. These application settings hold the name of the Windows Azure storage account and the account’s primary key. Change these settings to match your Windows Azure storage account name and storage account primary key, respectively.
     2. **NodesGroup**. This application setting holds the name of the Windows Azure node group. If you have a different group that contains Windows Azure worker nodes and would prefer to use it for this sample, change the value of this setting to the name of your group as it is specified in your HPC cluster’s node configuration.
     3. **HpcWebServiceUrl**. This application setting contains a URL that points to the location of the HPC Job Scheduler’s HTTP web service. For example, if your HPC Job Scheduler’s machine name is **MyHeadNode**, and the HPC cluster’s name is **MyCluster**, then the application setting will be:
        1. XML
        2. <add key="HpcWebServiceUrl"
        3. value="https://MyHeadNode/WindowsHpc/MyCluster"/>
  2. In the **Solution Explorer** window, expand the **HpcApplication** folder, expand the **AzureBlobCopy** project, and open the ***App.config*** file. The **appSettings** section contains two keys: ***StorageAccountName*** and ***StorageKey***. Change the values of these keys to match your Windows Azure storage account name and storage account primary key, respectively.
  3. In the **HpcApplication** folder, expand the **UploadToBOV** folder, and open the **app.config** file. The **appSettings** section contains two keys: ***StorageAccountName*** and ***StorageKey***. Change the values of these keys to match your Windows Azure storage account name and storage account primary key, respectively.
  4. Save all the changed files and build the solution.

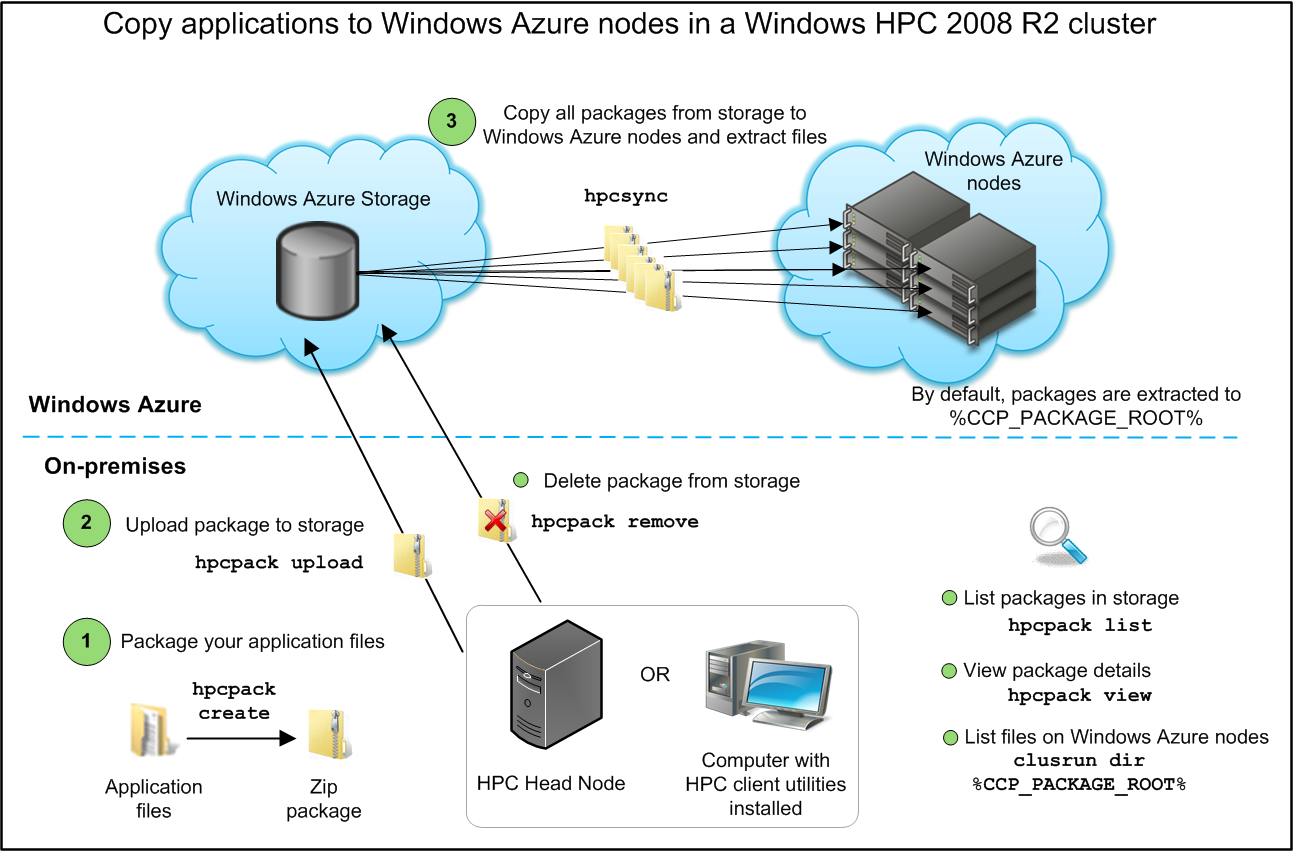
Task 4 - Uploading Input Files to Blob Storage

In this task, you will upload the input files required by the parametric sweep application to a blob in your Windows Azure storage account.

Uploading, downloading, and browsing files in blobs is an easy task if you install one of the blob storage browsing applications, such as [CloudBerry Explorer for Azure Blob Storage](http://cloudberrylab.com/?page=explorer-azure), or the [Azure Storage Explorer](http://azurestorageexplorer.codeplex.com). The following steps are for the CloudBerry Explorer application; you can use the same techniques with Azure Storage Explorer, but the steps may differ.

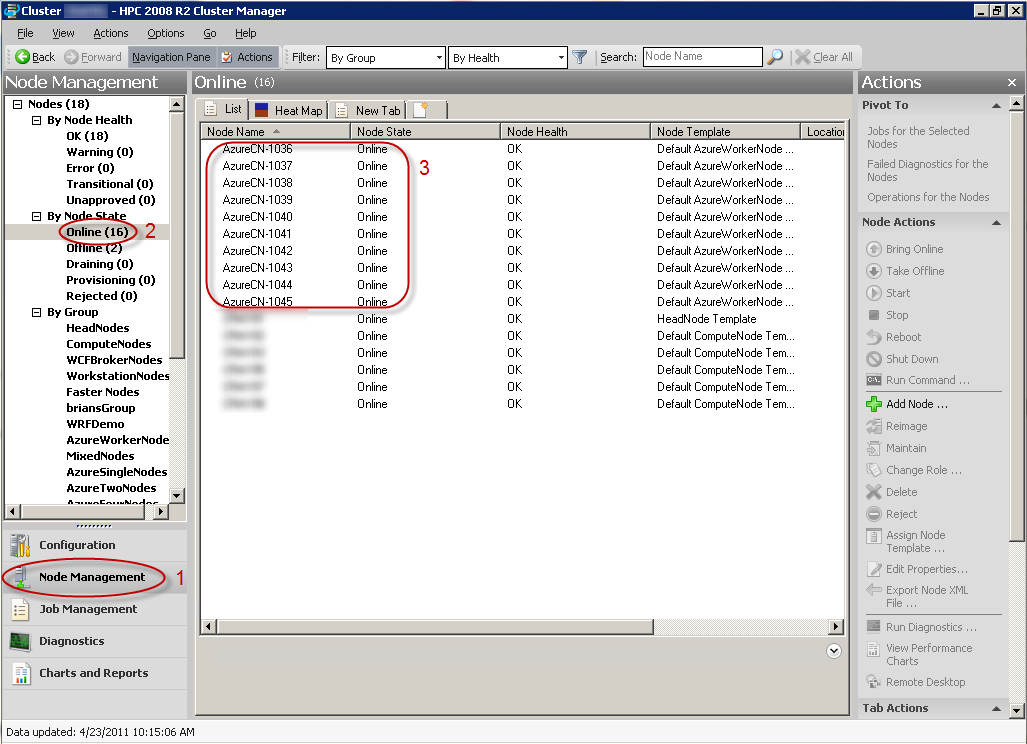
* 1. Download and install CloudBerry Explorer for Azure Blob Storage.
  2. Download and extract the human genome database files from the NCBI FTP server:
     1. Browse to the NCBI FTP server on <ftp://ftp.ncbi.nih.gov/blast/db>
     2. Download the files named **est\_human.00.tar.gz**, and **est\_human.01.tar.gz**.
     3. Extract both .gz files to the same folder. After extracting both files, you should have 17 files, as shown in Figure 3:
     4. 
     5. Figure 3
     6. Content of the extracted NCBI human genome database
  3. Open CloudBerry Explorer for Azure Blob Storage from **Start** | **All Programs** | **CloudBerryLab** | **CloudBerry Explorer for Azure Blob Storage** | **CloudBerry Explorer for Azure Blob Storage**.
  4. Open the **File** menu and select **Azure Blob Storage Accounts**. The **Account Registration** dialog will appear, as shown in Figure 4:
     1. 
     2. Figure 4
     3. The Account Registration dialog
  5. Click the **Add** button, and input the display name of the storage account, the storage account name, and the shared key (the primary access key) of the account. Use the same storage account settings you used in Task 3 for the **AzureBlobCopy** project.
  6. Click the **Test Connection** button and wait for the approval message. Close the approval message, click OK to add the storage account, and close the **Azure Blob Storage Accounts** dialog.
  7. You should now see your blob storage in the left pane of the application, and your machine (“My Computer”) in the right pane.
  8. Create a new container in the blob by clicking on the **New Container** button in the left pane, as shown in Figure 5:
     1. 
     2. Figure 5
     3. Creating a new blob container
  9. In the **Create New Container** dialog, set the container name to ***inputncbi***, select the **Full public read access** option from the Access control options as shown in Figure 6, and click **OK**.
     1. 
     2. Figure 6
     3. The Create New Container dialog
  10. Locate the newly created container in the list of containers and double-click its name to view its contents (it should be empty for now).
  11. In the right pane, navigate to the **Source\Input** folder that is in the **BLAST** sample folder.
  12. Select all the files (200 files) from the **Input** folder and click the **Copy** button. Click **Yes** in the confirmation message that appears, and then wait for the copy procedure to complete.
  13. Repeat steps 8 and 9 to create another new container, this time naming it ***ncbi***.
  14. In the left pane, click the **Root** folder in the address path to move to the root of the blob.
  15. Locate the ***ncbi***container in the list of containers and double-click its name to see its content (it should be empty for now).
  16. In the right pane, navigate to the folder to which you extracted the database files in step 2 of this task.
  17. Select all the extracted database files (17 files) from the database folder and click the **Copy** button. Click **Yes** in the confirmation message that appears, and then wait for the copy procedure to complete.
      1. **Note:** The size of the database is about 2.5GB, so this operation may take some time, depending on your network bandwidth.
  18. After the upload completes, close the CloudBerry Explorer application.

Deployment

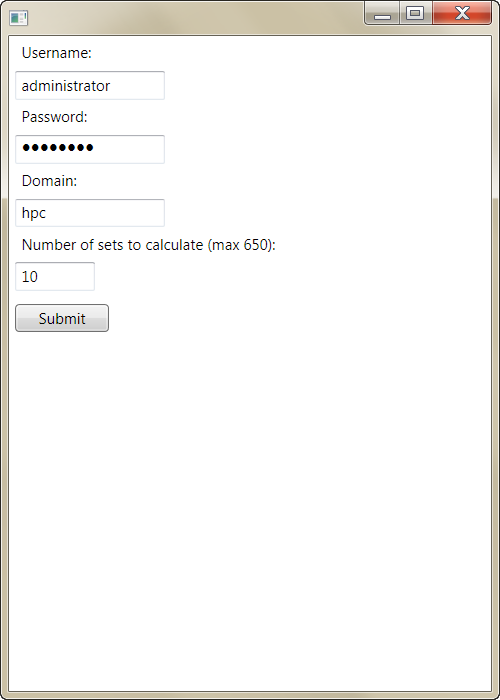
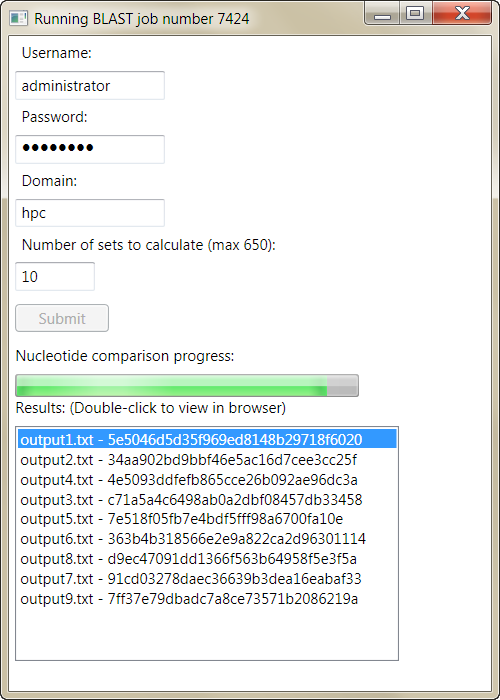
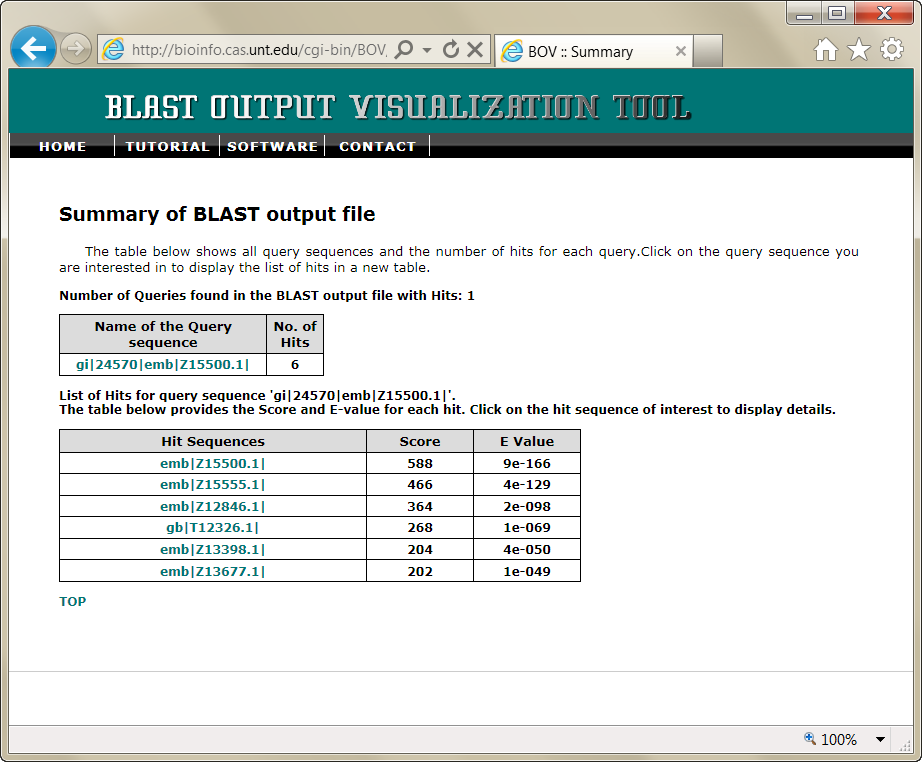
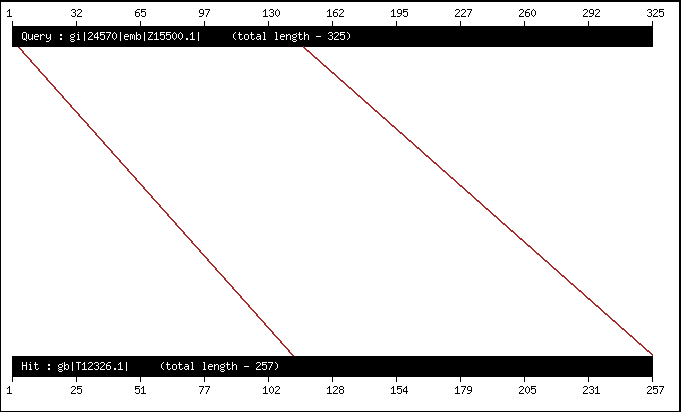
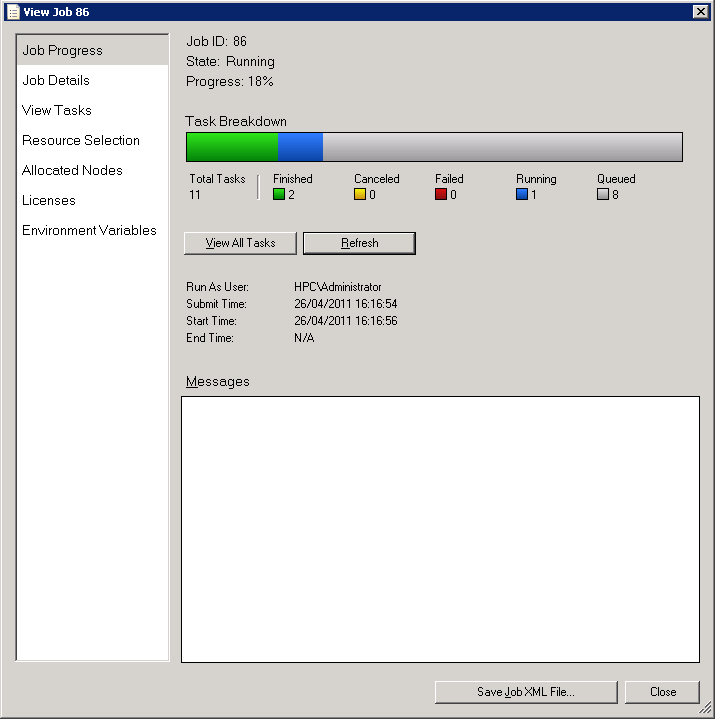
* 1. In order to run this sample, you will need to deploy the parametric sweep application to Windows Azure. To copy applications to each of the Windows Azure nodes, you will use the [hpcpack](http://technet.microsoft.com/en-us/library/gg481764(WS.10).aspx) command. This command can be used for packaging applications, uploading them to Windows Azure storage, and deploying packages to every Windows Azure worker node in the cluster, as shown in Figure 7:
  2. 
  3. Figure 7
  4. Using the hpcpack command

1. Task 1 – Deploying the Parametric Sweep Application to Windows Azure Nodes
   1. In this task, you will deploy the BLAST parametric sweep application to your Windows Azure nodes.
   2. Open the command prompt window from **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Visual Studio Tools** | **Visual Studio Command Prompt (2010)**.
   3. Navigate to the **BLAST** labs folder and run the following command to package the BLAST application so it can be deployment to the Windows Azure package storage (this step is illustrated in step 1 of Figure 7):
      1. CMD
         1. hpcpack create ncbi-blast-2.2.25.zip Source\ncbi-blast-2.2.25+
   4. Run the following command to upload the deployment package to the Windows Azure package storage (this step is illustrated in step 2 of Figure 7):
      1. CMD
         1. hpcpack upload ncbi-blast-2.2.25.zip /nodetemplate:"Azure node template" /relativePath:ncbi-blast-2.2.25
      2. **Note:** Change the value of the **nodetemplate** parameter to the name of your Windows Azure node template.
   5. If you already have Windows Azure nodes started in the HPC cluster, you need to copy the new package to them. To sync the Windows Azure nodes with the new packages stored in the application packages blob, run the following command (this step is illustrated in step 3 of Figure 7):
      1. CMD
         1. clusrun /nodegroup:WindowsAzureNodes hpcsync

Running the Client

1. Task 1 – Verifying Cluster State
   1. In this task, you will verify that the nodes in your cluster are online.
   2. Open the HPC 2008 R2 Cluster Manager application from **Start** | **All Programs** | **Microsoft HPC Pack 2008 R2** | **HPC Cluster Manager**.
   3. In the Cluster Manager application, enter the **Node Management** section and verify that the Windows Azure nodes in the cluster are online, as shown in Figure 8:
      1. 
      2. Figure 8
      3. Verifying the state of the Windows Azure nodes
2. Task 2 - Running the Client Application

In this task, you will run the WPF client application that submits the parametric sweep job and monitors its state.

* 1. Open Microsoft Visual Studio 2010 from **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Microsoft Visual Studio 2010**.
  2. Open the ***Blast.sln*** solution file located in the ***Blast\Source\Blast*** folder.
  3. Expand the **Client** folder, right-click the **BlastClient** project, and select **Set as StartUp Project**.
  4. From the **Debug** menu, select **Start Without Debugging** and wait for the WPF application to start and display the screen shown in Figure 9:
     1. 
     2. Figure 9
     3. The BLAST client application
  5. Enter the user name, password, and domain of an administrative user in the HPC cluster, select the number of nucleotides to match, and click the **Submit** button.
  6. After the job submits to the HPC Job Scheduler, a confirmation message will show. Click OK to close it.
  7. While the job is running, you will see a progress bar showing the progress of the job and a list of files (represented as file names and hash keys) that have been uploaded to the BOV website, as shown in Figure 10:
     1. 
     2. Figure 10
     3. Monitoring the progress of the parametric sweep job
  8. You can double-click on any of the hash keys to see the BOV web page with information about the selected nucleotide matches, as shown in Figure 11:
     1. 
     2. Figure 11
     3. The BOV summary page for the BLAST output file
  9. Click any of the matches and follow the instructions in the BOV web page to see a graphical representation of the match, as shown in Figure 12:
     + 1. 
       2. Figure 12
       3. High-scoring pair image for the nucleotide match
  10. While you wait for the application to finish running, you can also inspect the status of the parametric sweep job using the HPC 2008 R2 Cluster Manager utility, as shown in Figure 13:
      1. 
      2. Figure 13
      3. Status of the parametric sweep job
  11. Wait a couple of minutes for the application to complete.
  12. After the job is completed, you can continue to view the matches through the BOV web page.

Summary

* 1. After running the BLAST sample, you should have learned the following:
  + How to work with Windows Azure blob storage.
  + How to work with Windows Azure table storage.
  + How to use the HPC Pack REST interface.
  + How to deploy Windows Azure web roles to a hosted service.
  + How to package an application for Windows Azure.
  + How to deploy an application to Windows Azure nodes.
  + How to submit a parametric sweep job from a WPF client application.
  + How to check a running job’s status.